Foundation of Cryptography (0368-4162-01), Lecture 7

MACs and Signatures

Iftach Haitner, Tel Aviv University

December 27, 2011

Section 1

Message Authentication Code (MAC)

Message Authentication Code (MAC)

Goal: message authentication.

Message Authentication Code (MAC)

Goal: message authentication.

Definition 1 (MAC)

A trippet of PPT's $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ such that

1. $\mathsf{Gen}(1^n)$ outputs a key $k \in \{0,1\}^*$
2. $\mathsf{Mac}(k, m)$ outputs a “tag" t
3. $\mathsf{Vrfy}(k, m, t)$ output 1 (YES) or 0 (NO)

Message Authentication Code (MAC)

Goal: message authentication.

Definition 1 (MAC)

A trippet of PPT's (Gen, Mac, Vrfy) such that

1. $\mathrm{Gen}(1^n)$ outputs a key $k \in \{0,1\}^*$
2. $\mathrm{Mac}(k, m)$ outputs a "tag" t
3. $\mathrm{Vrfy}(k, m, t)$ output 1 (YES) or 0 (NO)

- **Consistency:** $\mathrm{Vrfy}(k, m, t) = 1$ for any $k \in \mathrm{Supp}(\mathrm{Gen}(1^n))$, $m \in \{0,1\}^n$ and $t = \mathrm{Mac}(k, m)$

Message Authentication Code (MAC)

Goal: message authentication.

Definition 1 (MAC)

A trippet of PPT's (Gen, Mac, Vrfy) such that

1. $\mathrm{Gen}(1^n)$ outputs a key $k \in \{0,1\}^*$
2. $\mathrm{Mac}(k, m)$ outputs a "tag" t
3. $\mathrm{Vrfy}(k, m, t)$ output 1 (YES) or 0 (NO)

- **Consistency:** $\mathrm{Vrfy}(k, m, t) = 1$ for any $k \in \mathrm{Supp}(\mathrm{Gen}(1^n))$, $m \in \{0,1\}^n$ and $t = \mathrm{Mac}(k, m)$
- **Unforgability:** For any oracle-aided PPT A:

$$\Pr\left[k \leftarrow \mathrm{Gen}(1^n); (m,t) \leftarrow \mathrm{A}^{\mathrm{Mac}_k, \mathrm{Vrfy}_k}(1^n): \mathrm{Vrfy}_k(m,t) = 1 \wedge \mathrm{Mac}_k \text{ was not asked on } m\right] = \mathrm{neg}(n)$$

where $\mathrm{Mac}_k(\cdot) := \mathrm{Mac}(k, \cdot)$ and $\mathrm{Vrfy}_k(\cdot) := \mathrm{Vrfy}(k, \cdot)$

- “Private key" definition

- “Private key" definition
- Definition too strong?

- “Private key" definition
- Definition too strong? Any message? Use of Verifier?

- “Private key" definition
- Definition too strong? Any message? Use of Verifier?
- “Replay attacks"

- “Private key" definition
- Definition too strong? Any message? Use of Verifier?
- “Replay attacks"
- Will focus on bounded length messages (specifically n), and then show how to move to any length

Bounded MACs

Definition 2 (ℓ-time MAC)

Same as in Definition 1, but security is only required against ℓ-query adversaries.

Zero-time MAC

Construction 3 (Zero-time MAC)

- $\mathsf{Gen}(1^n)$: outputs $k \leftarrow \{0,1\}^n$
- $\mathsf{Mac}_k(m) = k$
- $\mathsf{Vrfy}_k(m,t) = 1$, iff $t = k$

ℓ-wise independent hash

Definition 4 (ℓ-wise independent)

A function family $\mathcal{H}$ from $\{0,1\}^n$ to $\{0,1\}^m$ is ℓ-wise independent, where $\ell \in \mathbb{N}$, if for every distinct $x_1, \ldots, x_\ell \in \{0,1\}^n$ and every $y_1, \ldots, y_\ell \in \{0,1\}^m$, it holds that $\Pr_{h \leftarrow \mathcal{H}}[h(x_1) = y_1 \wedge \cdots \wedge h(x_\ell) = y_\ell] = 2^{-\ell m}$.

ℓ-times MAC

Construction 5 (ℓ-time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ be an efficient $(\ell+1)$-wise independent function family.

- $\mathsf{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\mathsf{Mac}(h, m) = h(m)$
- $\mathsf{Vrfy}(h, m, t) = 1$, iff $t = h(m)$

ℓ-times MAC

Construction 5 (ℓ-time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ be an efficient $(\ell+1)$-wise independent function family.

- $\mathsf{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\mathsf{Mac}(h, m) = h(m)$
- $\mathsf{Vrfy}(h, m, t) = 1$, iff $t = h(m)$

Claim 6

The above scheme is a (bounded length messages) ℓ-time MAC

ℓ-times MAC

Construction 5 (ℓ-time MAC)

Let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ be an efficient $(\ell+1)$-wise independent function family.

- $\mathsf{Gen}(1^n)$: outputs $h \leftarrow \mathcal{H}_n$
- $\mathsf{Mac}(h,m) = h(m)$
- $\mathsf{Vrfy}(h,m,t) = 1$, iff $t = h(m)$

Claim 6

The above scheme is a (bounded length messages) ℓ-time MAC

Proof: HW

OWF $\implies$ MAC

Construction 7

Same as Construction 5, but uses function $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

Claim 8

Assuming that $\mathcal{F}$ is a PRF, then Construction 7 is a MAC.

OWF $\implies$ MAC

Construction 7

Same as Construction 5, but uses function $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

Claim 8

Assuming that $\mathcal{F}$ is a PRF, then Construction 7 is a MAC.

Proof:

OWF $\implies$ MAC

Construction 7

Same as Construction 5, but uses function $\mathcal{F} = \{\mathcal{F}_n \colon \{0,1\}^n \mapsto \{0,1\}^n\}$ instead of $\mathcal{H}$.

Claim 8

Assuming that $\mathcal{F}$ is a PRF, then Construction 7 is a MAC.

Proof: Easy to prove if $\mathcal{F}$ is a family of random functions. Hence, also holds in case $\mathcal{F}$ is a PRF. $\square$

Collision Resistant Hash Family

Definition 9 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr[h \leftarrow \mathcal{H}_n, (x, x') \leftarrow \mathsf{A}(1^n, h) \colon x \neq x' \in \{0,1\}^* \\ \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

Collision Resistant Hash Family

Definition 9 (collision resistant hash family (CRH))

A function family $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ is collision resistant, if

$$\Pr[h \leftarrow \mathcal{H}_n, (x, x') \leftarrow \mathsf{A}(1^n, h) \colon x \neq x' \in \{0,1\}^* \\ \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

for any PPT A.

- Not known to be implied by OWF

Length restricted MAC $\implies$ MAC

Construction 10 (Length restricted MAC $\implies$ MAC)

Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be length restricted MAC with $d(n) = n$, and let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ be an eff. function family.

- $\mathsf{Gen}'(1^n)$: $k \leftarrow \mathsf{Gen}(1^n)$, $h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\mathsf{Mac}'_{k,h}(m) = \mathsf{Mac}_k(h(m))$
- $\mathsf{Vrfy}'_{k,h}(t, m) = \mathsf{Vrfy}_k(t, h(m))$

Length restricted MAC $\Longrightarrow$ MAC

Construction 10 (Length restricted MAC $\Longrightarrow$ MAC)

Let (Gen, Mac, Vrfy) be length restricted MAC with $d(n) = n$, and let $\mathcal{H} = \{\mathcal{H}_n : \{0,1\}^* \mapsto \{0,1\}^n\}$ be an eff. function family.

- $\mathrm{Gen}'(1^n)$: $k \leftarrow \mathrm{Gen}(1^n)$, $h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\mathrm{Mac}'_{k,h}(m) = \mathrm{Mac}_k(h(m))$
- $\mathrm{Vrfy}'_{k,h}(t, m) = \mathrm{Vrfy}_k(t, h(m))$

Claim 11

Assume $\mathcal{H}$ is an efficient collision resistant family, then $(\mathrm{Gen}', \mathrm{Mac}', \mathrm{Vrfy}')$ is a MAC.

Length restricted MAC $\implies$ MAC

Construction 10 (Length restricted MAC $\implies$ MAC)

Let $(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be length restricted MAC with $d(n) = n$, and let $\mathcal{H} = \{\mathcal{H}_n \colon \{0,1\}^* \mapsto \{0,1\}^n\}$ be an eff. function family.

- $\mathsf{Gen}'(1^n)$: $k \leftarrow \mathsf{Gen}(1^n)$, $h \leftarrow \mathcal{H}_n$. Set $k' = (k, h)$
- $\mathsf{Mac}'_{k,h}(m) = \mathsf{Mac}_k(h(m))$
- $\mathsf{Vrfy}'_{k,h}(t, m) = \mathsf{Vrfy}_k(t, h(m))$

Claim 11

Assume $\mathcal{H}$ is an efficient collision resistant family, then $(\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ is a MAC.

Proof: ?

Section 2

Signature Schemes

Definition

Definition 12 (Signature schemes)

A trippet of PPT's (Gen, Sign, Vrfy) such that

1. $\mathrm{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathrm{Sign}(s, m)$ outputs a "signature" $\sigma \in \{0,1\}^*$
3. $\mathrm{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO)

Definition

Definition 12 (Signature schemes)

A trippet of PPT's (Gen, Sign, Vrfy) such that

1. $\mathrm{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathrm{Sign}(s, m)$ outputs a "signature" $\sigma \in \{0,1\}^*$
3. $\mathrm{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO)

- **Consistency:** $\mathrm{Vrfy}(v, m, \sigma) = 1$ for any $(s, v) \in \mathrm{Supp}(\mathrm{Gen}(1^n))$, $m \in \{0,1\}^*$ and $\sigma \in \mathrm{Supp}(\mathrm{Sign}(s, m))$

Definition

Definition 12 (Signature schemes)

A trippet of PPT's (Gen, Sign, Vrfy) such that

1. $\mathrm{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathrm{Sign}(s, m)$ outputs a “signature" $\sigma \in \{0,1\}^*$
3. $\mathrm{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO)

- **Consistency:** $\mathrm{Vrfy}(v, m, \sigma) = 1$ for any $(s, v) \in \mathrm{Supp}(\mathrm{Gen}(1^n))$, $m \in \{0,1\}^*$ and $\sigma \in \mathrm{Supp}(\mathrm{Sign}(s, m))$
- **Unforgability:** For any oracle-aided PPT A

$$\Pr\Big[(s, v) \leftarrow \mathrm{Gen}(1^n); (m, \sigma) \leftarrow \mathrm{A}^{\mathrm{Sign}_s}(1^n, v) : \\ \mathrm{Vrfy}_v(m, \sigma) = 1 \land \mathrm{Sign}_s \text{ was not asked on } m\Big] = \mathrm{neg}(n)$$

where $\mathrm{Sign}_s(\cdot) := \mathrm{Sign}(s, \cdot)$

Definition

Definition 12 (Signature schemes)

A trippet of PPT's (Gen, Sign, Vrfy) such that

1. $\mathrm{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathrm{Sign}(s, m)$ outputs a "signature" $\sigma \in \{0,1\}^*$
3. $\mathrm{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO)

- **Consistency:** $\mathrm{Vrfy}(v, m, \sigma) = 1$ for any $(s, v) \in \mathrm{Supp}(\mathrm{Gen}(1^n))$, $m \in \{0,1\}^*$ and $\sigma \in \mathrm{Supp}(\mathrm{Sign}(s, m))$
- **Unforgability:** For any oracle-aided PPT A

$$\Pr\Big[(s, v) \leftarrow \mathrm{Gen}(1^n); (m, \sigma) \leftarrow \mathrm{A}^{\mathrm{Sign}_s}(1^n, v) : \mathrm{Vrfy}_v(m, \sigma) = 1 \land \mathrm{Sign}_s \text{ was not asked on } m\Big] = \mathrm{neg}(n)$$

where $\mathrm{Sign}_s(\cdot) := \mathrm{Sign}(s, \cdot)$ (similarly $\mathrm{Vrfy}_v(\cdot) := \mathrm{Vrfy}(v, \cdot)$)

Definition

Definition 12 (Signature schemes)

A trippet of PPT's (Gen, Sign, Vrfy) such that

1. $\mathsf{Gen}(1^n)$ outputs a pair of keys $(s, v) \in \{0,1\}^* \times \{0,1\}^*$
2. $\mathsf{Sign}(s, m)$ outputs a "signature" $\sigma \in \{0,1\}^*$
3. $\mathsf{Vrfy}(v, m, \sigma)$ outputs 1 (YES) or 0 (NO)

- **Consistency:** $\mathsf{Vrfy}(v, m, \sigma) = 1$ for any $(s, v) \in \mathsf{Supp}(\mathsf{Gen}(1^n))$, $m \in \{0,1\}^*$ and $\sigma \in \mathsf{Supp}(\mathsf{Sign}(s, m))$
- **Unforgability:** For any oracle-aided PPT A

$$\Pr\Big[(s, v) \leftarrow \mathsf{Gen}(1^n); (m, \sigma) \leftarrow \mathsf{A}^{\mathsf{Sign}_s}(1^n, v) : \\ \mathsf{Vrfy}_v(m, \sigma) = 1 \land \mathsf{Sign}_s \text{ was not asked on } m\Big] = \mathsf{neg}(n)$$

where $\mathsf{Sign}_s(\cdot) := \mathsf{Sign}(s, \cdot)$ (similarly $\mathsf{Vrfy}_v(\cdot) := \mathsf{Vrfy}(v, \cdot)$)

- Signature $\implies$ MAC

- Signature $\implies$ MAC
- “Harder" to construct than MACs: (even restricted forms) require OWF

- Signature $\implies$ MAC
- “Harder" to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given

- Signature $\implies$ MAC
- “Harder" to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given
- Strong signatures: infeasible to generate *any* new valid signatures (even for message for which a signature was asked)

- Signature $\implies$ MAC
- “Harder" to construct than MACs: (even restricted forms) require OWF
- Oracle access to Vrfy is not given
- Strong signatures: infeasible to generate *any* new valid signatures (even for message for which a signature was asked)

Theorem 13

OWFs imply strong signatures.

Section 3

OWFs $\implies$ Signatures

One Time Signatures

Definition 14 (one time signatures)

Same as Definition 12, but A can only ask for a single signature.

One Time Signatures

Definition 14 (one time signatures)

Same as Definition 12, but A can only ask for a single signature.

Definition 15 (length restricted, one time signatures)

Same as Definition 14, but A can only ask for signatures of predetermined length (in our case n).

One Time Signatures

Definition 14 (one time signatures)

Same as Definition 12, but A can only ask for a single signature.

Definition 15 (length restricted, one time signatures)

Same as Definition 14, but A can only ask for signatures of predetermined length (in our case n).

- Assuming CRH exists: length restricted, one time signatures $\Longrightarrow$ one time signatures.

OWF $\Longrightarrow$ length restricted, One Time Signature

Construction 16 (length restricted, one time signature)

Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$.

1. Gen(1^n): $s_1^0, s_1^1, \ldots, s_n^0, s_n^1 \leftarrow \{0,1\}^n$, let $s = \left(s_1^0, s_1^1, \ldots, s_n^0, s_n^1\right)$ and $v = \left(v_1^0 = f(s_1^0), v_1^1 = f(s_1^1), \ldots, v_n^0 = f(s_n^0), v_n^1 = f(s_n^1)\right)$
2. Sign(s, m): Output $(s_1^{m_1}, \ldots, s_n^{m_n})$
3. Vrfy($v, m, \sigma = (\sigma_1, \ldots, \sigma_n)$) check that $f(\sigma_i) = v_{m_i}$ for all $i \in [n]$

OWF $\Longrightarrow$ length restricted, One Time Signature

Construction 16 (length restricted, one time signature)

Let $f\colon \{0,1\}^n \mapsto \{0,1\}^n$.

1. $\mathrm{Gen}(1^n)$: $s_1^0, s_1^1, \ldots, s_n^0, s_n^1 \leftarrow \{0,1\}^n$, let $s = \left(s_1^0, s_1^1, \ldots, s_n^0, s_n^1\right)$ and $v = \left(v_1^0 = f(s_1^0), v_1^1 = f(s_1^1), \ldots, v_n^0 = f(s_n^0), v_n^1 = f(s_n^1)\right)$
2. $\mathrm{Sign}(s, m)$: Output $(s_1^{m_1}, \ldots, s_n^{m_n})$
3. $\mathrm{Vrfy}(v, m, \sigma = (\sigma_1, \ldots, \sigma_n))$ check that $f(\sigma_i) = v_{m_i}$ for all $i \in [n]$

Lemma 17

Assume that f is a OWF, then scheme from Construction 16 is a length restricted one-time signature scheme

Proving Lemma 17

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 16, we use A to invert f.

Algorithm 18 (Inv)

Input: $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$, with y.
2. If A$(1^n, v)$ asks to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
3. Let (m, σ) be A's output. If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

Proving Lemma 17

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 16, we use A to invert f.

Algorithm 18 (Inv)

Input: $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$, with y.
2. If $A(1^n, v)$ asks to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
3. Let (m, σ) be A's output. If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

v is distributed as it is in the real “signature game" (ind. of i^* and j^*).

Proving Lemma 17

Let a PPT A, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that break the security of Construction 16, we use A to invert f.

Algorithm 18 (Inv)

Input: $y \in \{0,1\}^n$

1. Choose $(s, v) \leftarrow Gen(1^n)$ and replace $v_{j^*}^{i^*}$ for a random $i^* \in [n]$ and $j^* \in \{0,1\}$, with y.
2. If $A(1^n, v)$ asks to sign message $m \in \{0,1\}^n$ with $m_{i^*} = j^*$ abort, otherwise use s to answer the query.
3. Let (m, σ) be A's output. If σ is not a valid signature for m, or $m_{i^*} \neq j^*$, abort.
   Otherwise, return σ_{i^*}.

v is distributed as it is in the real "signature game" (ind. of i^* and j^*). Therefore Inv inverts f w.p. $\frac{1}{2np(n)}$ for any $n \in \mathcal{I}$.

Stateful schemes (also known as, Memory-dependant schemes)

Definition 19 (Stateful scheme)

Same as in Definition 12, but Sign might keep state.

Stateful schemes (also known as, Memory-dependant schemes)

Definition 19 (Stateful scheme)

Same as in Definition 12, but Sign might keep state.

- Make sense in many applications (e.g., , smartcards)

Stateful schemes (also known as, Memory-dependant schemes)

Definition 19 (Stateful scheme)

Same as in Definition 12, but Sign might keep state.

- Make sense in many applications (e.g., , smartcards)
- We'll use it a building block for building a stateless scheme

Naive construction

Let (Gen, Sign, Vrfy) be a one-time signature scheme.

Construction 20 (Naive construction)

1. $\mathrm{Gen}'(1^n)$ outputs $(s_1, v_1) = \mathrm{Gen}(1^n)$.
2. $\mathrm{Sign}_s(m_i)$, where m_i is i'th message to sign:
   Let $((m_1, \sigma'_1), \ldots, (m_{i-1}, \sigma'_{i-1}))$ be the previously signed pairs of messages/signatures.
   1. Let $(s_{i+1}, v_{i+1}) \leftarrow \mathrm{Gen}(1^n)$
   2. Let $\sigma_i = \mathrm{Sign}_{s_{i-1}}(m_i, v_{i+1})$, and output $\sigma'_i = (\sigma'_{i-1}, m_i, v_{i+1}, \sigma_i)$.
3. $\mathrm{Vrfy}'_v(m, \sigma' = (m_1, v_2, \sigma_1), \ldots, (m_i, v_{i+1}, \sigma_i))$:
   1. Verify $\mathrm{Vrfy}_{v_{j-1}}((m_j, v_{j+1}), \sigma_j) = 1$ for every $j \in [i]$
   2. Verify $m_i = m$

1. State is used for maintaining the private key (e.g., s_i') and to prevent using the same one-time signature twice.
2. Inefficient scheme, thought still polynomial, both running time and signature size are linear in number of signatures

1. State is used for maintaining the private key (e.g., s_i') and to prevent using the same one-time signature twice.
2. Inefficient scheme, thought still polynomial, both running time and signature size are linear in number of signatures
3. Critically uses the fact that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ is works for any length

Lemma 21

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful signature scheme.*

Lemma 21

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful signature scheme.*

Proof: Let a PPT A', $\mathcal{I} \subseteq \mathbb{N}$ and $p \in \mathrm{poly}$ that breaks the security of $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$, we present a PPT A that breaks the security of $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$.

Lemma 21

Assume that (Gen, Sign, Vrfy) *is one time signature scheme, then* (Gen′, Sign′, Vrfy′) *is a stateful signature scheme.*

Proof: Let a PPT A′, $\mathcal{I} \subseteq \mathbb{N}$ and $p \in$ poly that breaks the security of (Gen′, Sign′, Vrfy′), we present a PPT A that breaks the security of (Gen, Sign, Vrfy).

- We assume for simplicity that p also bounds the query complexity of A′

Proving Lemma 21 cont.

Let the random variables
$(m, \sigma = (m_1, v_2, \sigma_1), \ldots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A′

Proving Lemma 21 cont.

Let the random variables
$(m, \sigma = (m_1, v_2, \sigma_1), \ldots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A′

Claim 22

Whenever A′ succeeds, $\exists \widetilde{i} = \widetilde{i}(m, \sigma) \in [q]$ such that:

1. Sign′ *was not* asked by A′ on $m_{\widetilde{i}}$.
2. Sign′ *was* asked by A′ on m_i, for every $i \in [\widetilde{i} - 1]$

Proving Lemma 21 cont.

Let the random variables
$(m, \sigma = (m_1, v_2, \sigma_1), \ldots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A′

Claim 22

Whenever A′ succeeds, $\exists \widetilde{i} = \widetilde{i}(m, \sigma) \in [q]$ such that:

1. Sign′ *was not* asked by A′ on $m_{\widetilde{i}}$.
2. Sign′ *was* asked by A′ on m_i, for every $i \in [\widetilde{i} - 1]$

Proof:

Proving Lemma 21 cont.

Let the random variables
$(m, \sigma = (m_1, v_2, \sigma_1), \ldots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A′

Claim 22

Whenever A′ succeeds, $\exists \widetilde{i} = \widetilde{i}(m, \sigma) \in [q]$ such that:

1. Sign′ *was not* asked by A′ on $m_{\widetilde{i}}$.
2. Sign′ *was* asked by A′ on m_i, for every $i \in [\widetilde{i} - 1]$

Proof: Let $\widetilde{i}$ be the maximal index such that condition (2) holds (cannot be $q + 1$).$\square$

Proving Lemma 21 cont.

Let the random variables
$(m, \sigma = (m_1, v_2, \sigma_1), \ldots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A′

Claim 22

Whenever A′ succeeds, $\exists \widetilde{i} = \widetilde{i}(m, \sigma) \in [q]$ such that:

1. Sign′ *was not* asked by A′ on $m_{\widetilde{i}}$.
2. Sign′ *was* asked by A′ on m_i, for every $i \in [\widetilde{i} - 1]$

Proof: Let $\widetilde{i}$ be the maximal index such that condition (2) holds (cannot be $q + 1$).$\square$

- Let $\widetilde{m} = (m_{\widetilde{i}}, v_{\widetilde{i}+1})$, and let $s_{\widetilde{i}}$ be the signing key generated together with $v_{\widetilde{i}}$.

Proving Lemma 21 cont.

Let the random variables
$(m, \sigma = (m_1, v_2, \sigma_1), \ldots, (m_q, v_{q+1}, \sigma_q))$ be the pair output by A′

Claim 22

Whenever A′ succeeds, $\exists \widetilde{i} = \widetilde{i}(m, \sigma) \in [q]$ such that:

1. Sign′ *was not* asked by A′ on $m_{\widetilde{i}}$.
2. Sign′ *was* asked by A′ on m_i, for every $i \in [\widetilde{i} - 1]$

Proof: Let $\widetilde{i}$ be the maximal index such that condition (2) holds (cannot be $q + 1$).$\square$

- Let $\widetilde{m} = (m_{\widetilde{i}}, v_{\widetilde{i}+1})$, and let $s_{\widetilde{i}}$ be the signing key generated together with $v_{\widetilde{i}}$.
- Hence, $\text{Sign}_{s_{\widetilde{i}}}(\sigma_{\widetilde{i}}, \widetilde{m}) = 1$, and Sign_{s_i} was not queried by Sign'_s on $\widetilde{m}$.

Definition of A

Algorithm 23 (A)

Input: v, 1^n
Oracle: Sign_s

1. Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \mathrm{Gen}'(1^n)$.
2. Emulate a random execution of $\mathsf{A}'^{\mathrm{Sign}'_{s'}}$ with a single twist:
   - On the i^*'th call to $\mathrm{Sign}'_{s'}$, set $v_{i^*} = v$ (rather then choosing it via Gen)
   - When need to sign using s_{i^*}, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q)) \leftarrow \mathsf{A}'$
4. Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > q$))

Definition of A

Algorithm 23 (A)

Input: v, 1^n
Oracle: Sign_s

1. Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \text{Gen}'(1^n)$.
2. Emulate a random execution of $\text{A}'^{\text{Sign}'_{s'}}$ with a single twist:
   - On the i^*'th call to $\text{Sign}'_{s'}$, set $v_{i^*} = v$ (rather then choosing it via Gen)
   - When need to sign using s_{i^*}, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q)) \leftarrow \text{A}'$
4. Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > q$))

- Sign_s is called at most once

Definition of A

Algorithm 23 (A)

Input: v, 1^n
Oracle: Sign_s

1. Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \text{Gen}'(1^n)$.
2. Emulate a random execution of $\text{A}'^{\text{Sign}'_{s'}}$ with a single twist:
   - On the i^*'th call to $\text{Sign}'_{s'}$, set $v_{i^*} = v$ (rather then choosing it via Gen)
   - When need to sign using s_{i^*}, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q)) \leftarrow \text{A}'$
4. Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > q$))

- Sign_s is called at most once
- The emulated game $\text{A}'^{\text{Sign}'_{s'}}$ has the “right" distribution.

Definition of A

Algorithm 23 (A)

Input: v, 1^n
Oracle: Sign_s

1. Choose $i^* \leftarrow [p = p(n)]$ and $(s', v') \leftarrow \mathsf{Gen}'(1^n)$.
2. Emulate a random execution of $\mathsf{A}'^{\mathsf{Sign}'_{s'}}$ with a single twist:
   - On the i^*'th call to $\mathsf{Sign}'_{s'}$, set $v_{i^*} = v$ (rather then choosing it via Gen)
   - When need to sign using s_{i^*}, use Sign_s.
3. Let $(m, \sigma = (m_1, v_1, \sigma_1), \ldots, (m_q, v_q, \sigma_q)) \leftarrow \mathsf{A}'$
4. Output $((m_{i^*}, v_{i^*}), \sigma_{i^*})$ (abort if $i^* > q$))

- Sign_s is called at most once
- The emulated game $\mathsf{A}'^{\mathsf{Sign}'_{s'}}$ has the “right" distribution.
- A breaks $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ whenever $i^* = \widetilde{i} > 1$.

Analysis of A

For any $n \in \mathcal{I}$

$$\begin{aligned} &\Pr[\mathsf{A}(1^n) \text{ breaks } (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})] \\ &\geq \quad \Pr_{i^* \leftarrow [p = p(n)]}[i = \widetilde{i}] \\ &\geq \quad \frac{1}{p} \cdot \Pr[\mathsf{A}' \text{ breaks } (\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')] \geq \frac{1}{p(n)^2} \end{aligned}$$

"Somewhat"-Stateful Schemes

A one-time scheme (Gen, Sign, Vrfy), and $\ell = \ell(n) \in \omega(\log n)$

Construction 24

- $\mathrm{Gen}'(1^n)$: output $(s_\lambda, v_\lambda) \leftarrow \mathrm{Gen}(1^n)$.
- $\mathrm{Sign}'_s(m)$: choose *unused* $\bar{r} \in \{0,1\}^\ell$
  1. For $i = 0$ to $\ell - 1$: if $a_{\bar{r}_{1,\ldots,i}}$ was not set:
     1. For both $j \in \{0,1\}$, let $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \mathrm{Gen}(1^n)$
     2. $\sigma_{\bar{r}_{1,\ldots,i}} = \mathrm{Sign}_{s_{\bar{r}_{1,\ldots,i}}}(a_{1,\ldots,i} = (v_{\bar{r}_{1,\ldots,i},0}, v_{\bar{r}_{1,\ldots,i},1}))$
  2. Output $(\bar{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\bar{r}_{1,\ldots,\ell-1}}, \sigma_{\bar{r}_{1,\ldots,\ell-1}}, \sigma_{\bar{r}} = \mathrm{Sign}_{s_{\bar{r}}}(m))$
- $\mathrm{Vrfy}'_v(m, \sigma' = (\bar{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\bar{r}-1}, \sigma_{\bar{r}_{1,\ldots,\ell-1}}, \sigma_{\bar{r}})$
  1. Verify $\mathrm{Vrfy}_{v_{\bar{r}_{1,\ldots,i}}}(a_{\bar{r}_{1,\ldots,i}}, \sigma_{\bar{r}_{1,\ldots,i}}) = 1$ for every $i \in \{0, \ldots, \ell - 1\}$
  2. Verify $\mathrm{Vrfy}_{v_{\bar{r}}}(m, \sigma_{\bar{r}}) = 1$ (where $v_{\bar{r}} = (a_{\bar{r}})_{\bar{r}[\ell]}$)

Somewhat-Stateful Schemes

1. More efficient scheme

1. More efficient scheme
2. Sign' does not keep track of the message history.

1. More efficient scheme
2. Sign' does not keep track of the message history.
3. Each leaf is visited at most once.

1. More efficient scheme
2. Sign$'$ does not keep track of the message history.
3. Each leaf is visited at most once.
4. Each one-time signature is used once.

Lemma 25

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is one time signature scheme, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *is a stateful signature scheme.*

Proof:

Lemma 25

Assume that $(\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ *is one time signature scheme, then* $(\mathrm{Gen}', \mathrm{Sign}', \mathrm{Vrfy}')$ *is a stateful signature scheme.*

Proof: Let $(m, \sigma' = (\bar{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\bar{r}-1}, \sigma_{\bar{r}_{1,\ldots,\ell-1}}, \sigma_{\bar{r}})$ be the output of a cheating A$'$ and let $a_{\bar{r}} = m$

Lemma 25

Assume that (Gen, Sign, Vrfy) *is one time signature scheme, then* (Gen′, Sign′, Vrfy′) *is a stateful signature scheme.*

Proof: Let $(m, \sigma' = (\overline{r}, a_\lambda, \sigma_\lambda, \ldots, a_{\overline{r}-1}, \sigma_{\overline{r}_{1,\ldots,\ell-1}}, \sigma_{\overline{r}})$ be the output of a cheating A′ and let $a_{\overline{r}} = m$

Claim 26

Whenever A′ succeeds, $\exists \widetilde{i} = \widetilde{i}(m, \sigma') \in \{0, \ldots, \ell\}$ such that:

1. Sign'_s queried $\mathrm{Sign}_{s_{\overline{r}_{1,\ldots,i}}}(a_{\overline{r}_{1,\ldots,i}})$ for every $i \in [\widetilde{i} - 1]$, where $s_{\overline{r}_{1,\ldots,i}}$ is the value sampled by Sign′ when sampling $a_{\overline{r}_{1,\ldots,i-1}}$ (or s_λ, if $\widetilde{i} = 0$)
2. Sign'_s did not query $\mathrm{Sign}_{s_{\overline{r}_{1,\ldots,i}}}(a_{\overline{r}_{1,\ldots,i}})$.

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

1. $\mathrm{Gen}'(1^n)$: let $(s,v) \leftarrow \mathrm{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in \mathrm{poly}$ is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

1. $\mathrm{Gen}'(1^n)$: let $(s, v) \leftarrow \mathrm{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in$ poly is large enough for the application below, and outputs $(s' = (s, \pi), v' = v)$
2. $\mathrm{Sign}'(1^n)$:
   1. choose $\bar{r} = \pi(0^\ell \circ m)_{1,\ldots,\ell}$

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

1. $\mathsf{Gen}'(1^n)$: let $(s,v) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in$ poly is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$
2. $\mathsf{Sign}'(1^n)$:
   1. choose $\bar{r} = \pi(0^\ell \circ m)_{1,\ldots,\ell}$
   2. When setting $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n)$, use $\pi(\bar{r}_{1,\ldots,i},j)$ as the randomness for Gen.

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

1. $\mathrm{Gen}'(1^n)$: let $(s,v) \leftarrow \mathrm{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in \mathrm{poly}$ is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$
2. $\mathrm{Sign}'(1^n)$:
    1. choose $\bar{r} = \pi(0^\ell \circ m)_{1,\ldots,\ell}$
    2. When setting $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \mathrm{Gen}(1^n)$, use $\pi(\bar{r}_{1,\ldots,i},j)$ as the randomness for Gen.

- Sign' keeps no state

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

1. $\text{Gen}'(1^n)$: let $(s,v) \leftarrow \text{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in$ poly is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$
2. $\text{Sign}'(1^n)$:
   1. choose $\bar{r} = \pi(0^\ell \circ m)_{1,\ldots,\ell}$
   2. When setting $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \text{Gen}(1^n)$, use $\pi(\bar{r}_{1,\ldots,i},j)$ as the randomness for Gen.

- Sign' keeps no state
- A single one-time signature key might be used several times, but always on *the same* message

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

1. $\mathsf{Gen}'(1^n)$: let $(s,v) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in$ poly is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$
2. $\mathsf{Sign}'(1^n)$:
   1. choose $\bar{r} = \pi(0^\ell \circ m)_{1,\ldots,\ell}$
   2. When setting $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n)$, use $\pi(\bar{r}_{1,\ldots,i},j)$ as the randomness for Gen.

- Sign' keeps no state
- A single one-time signature key might be used several times, but always on *the same* message

Efficient scheme:

Stateless Scheme

Inefficient scheme:

Let $\Pi_{\ell,q}$ be the set of random functions from $\{0,1\}^*$ to $\{0,1\}^q$.

1. $\mathsf{Gen}'(1^n)$: let $(s,v) \leftarrow \mathsf{Gen}(1^n)$ and $\pi \leftarrow \Pi_{\ell(n),q(n)}$, where $q \in$ poly is large enough for the application below, and outputs $(s' = (s,\pi), v' = v)$
2. $\mathsf{Sign}'(1^n)$:
    1. choose $\bar{r} = \pi(0^\ell \circ m)_{1,\ldots,\ell}$
    2. When setting $(s_{\bar{r}_{1,\ldots,i},j}, v_{\bar{r}_{1,\ldots,i},j}) \leftarrow \mathsf{Gen}(1^n)$, use $\pi(\bar{r}_{1,\ldots,i},j)$ as the randomness for Gen.

- Sign' keeps no state
- A single one-time signature key might be used several times, but always on *the same* message

Efficient scheme: use PRF

Without CRH

Definition 27 (target collision resistant (TCR))

A function family $\mathcal{H} = \{\mathcal{H}_n\}$ is target collision resistant, if any pair of PPT's $\mathsf{A}_1, \mathsf{A}_2$:

$$\Pr[(x, a) \leftarrow \mathsf{A}_1(1^n); h \leftarrow \mathcal{H}_n; x' \leftarrow \mathsf{A}_2(a, h) : \\ x \neq x' \wedge h(x) = h(x')] = \mathrm{neg}(n)$$

Without CRH

Definition 27 (target collision resistant (TCR))

A function family $\mathcal{H} = \{\mathcal{H}_n\}$ is target collision resistant, if any pair of PPT's $\mathsf{A}_1, \mathsf{A}_2$:

$$\Pr[(x, a) \leftarrow \mathsf{A}_1(1^n); h \leftarrow \mathcal{H}_n; x' \leftarrow \mathsf{A}_2(a, h) : x \neq x' \wedge h(x) = h(x')] = \operatorname{neg}(n)$$

Theorem 28

OWFs imply efficient compressing TCRs.

Definition 29 (target one-time signatures)

A trippet of PPT's (Gen, Sign, Vrfy) is target one-time signatures (TOS), if

- **Consistency:** same as in Definition 14
- **Target unforgability:** for any pair of PPT's A_1, A_2

$$\Pr\left[(m,a) \leftarrow A_1(1^n); (s,v) \leftarrow \mathrm{Gen}(1^n); (m',\sigma) \leftarrow A(a, \mathrm{Sign}_s(m)) : m' \neq m \wedge \mathrm{Vrfy}_v(m',\sigma) = 1\right] = \mathrm{neg}(n)$$

Definition 29 (target one-time signatures)

A trippet of PPT's (Gen, Sign, Vrfy) is target one-time signatures (TOS), if

- **Consistency:** same as in Definition 14
- **Target unforgability:** for any pair of PPT's A_1, A_2

$$\Pr\Big[(m,a) \leftarrow A_1(1^n); (s,v) \leftarrow \mathrm{Gen}(1^n);$$
$$(m',\sigma) \leftarrow A(a, \mathrm{Sign}_s(m))\colon m' \neq m \wedge \mathrm{Vrfy}_v(m',\sigma) = 1\Big]$$
$$= \mathrm{neg}(n)$$

Claim 30

OWFs imply target one-time signatures

Lemma 31

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is TOS, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *from Construction 24 is a stateful signature scheme.*

Lemma 31

Assume that $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *is TOS, then* $(\mathsf{Gen}', \mathsf{Sign}', \mathsf{Vrfy}')$ *from Construction 24 is a stateful signature scheme.*

Proof: ?